**기업연계 멘토링 진행(4차)**

|  |  |  |  |
| --- | --- | --- | --- |
| **지역/반** | 서울 2반 | **일자/시간** | 21.05.04 14시 |
| **팀장** | 최낙훈 | **컨설턴트** | 이태희 |
| **기업명** | 삼성 SDS | **주제명** | 테스트 자동화 |
| **세부 내용** | | | |
| **[참여인원]**  하성민 프로님(SDS 기업 담당 프로님),  A201팀 (팀장 최낙훈, 팀원 김영록, 백현오, 이병희, 전원표)  박용미 교육프로님  이태희 컨설턴트님, 신채원 실습코치님  **[주요 진행 내용]**  3주차 멘토링 이후 프로젝트 진행 상황 설명 → 발표 자료 피드백  프로젝트 관련 질문 리스트 답변  **[질문/답변 내용]**  **1. 기존 서비스를 이용하면서 불편했던 점이나 개선되었으면 하는 점 있으신가요?**  - 개인화된 리포트  필터링 되어진 리포트를 보고 싶었습니다. 정리해서 보여줄 수 있는 페이지를 보여줄 수 있으면 좋지 않을까 생각 했었습니다.  **2. 경영진(높은 분들)이 볼만한 Dashboard를 만드려고 합니다. 저희가 생각한 기능은 시스템에 등록된 모든 테스트 결과의 평균치, (일, 주, 월)간 테스트 결과의 평균치 , Pass율 혹은 Coverage가 가장 (높은, 낮은) 프로젝트를 진행중인 팀 표시 입니다. 이외에도 들어가면 좋을 것 같은 기능이 무엇이 있을까요??**  - 경영진이 주로 사용한다고 하면 평가도 좋지만, 개발팀에 도움을 줄 수 있는 정보를 제공하는 것이 좋습니다.  - 시스템의 개념을 작게 보는 것 같습니다  - 기준점을 삼아 경고 표시를 하는 정도가 좋지 않을까 생각합니다.  - 마지막 테스트/빌드 일정이 언제인지도 알 수 있으면 좋습니다.  - 표에서 수(숫자)를 %(퍼센트)로 표현하는 것이 개발자 입장에서 더 보기 편할 것입니다. 개발자는 숫자보다는 %로 봤을 때 느끼는 점이 더 큽니다.  - 테스트, 커버리지의 history 기간을 어느 정도의 기준으로 잡아야 할 지 고민해보셔야 할 것 입니다.  **3. Testopia에서는 Project를 조회하는 페이지, TestJob을 조회하는 페이지가 존재하던데 Project 페이지와 TestJob 페이지를 나눈 특별한 이유가 있을까요??**  - 어떤 것을 중심으로 시스템을 볼 것이냐의 차이입니다. (Test결과를 Job을 기준으로 받을 건지, 프로젝트를 기준으로 받을 건지)일단 나눈 이유는 프로젝트가 필수가 아니라서 분리한 것입니다. 프로젝트는 말 그대로 껍데기이고, 저희(삼성SDS 서비스)는 job을 기준으로 생각하여 job id를 미리 지정해서 데이터를 저장합니다**.**  **4. Jacoco 결과를 파싱을 하면 다양한 depth의 정보를 가져올 수 있는데 어디까지의 데이터를 표현해주면 좋을 지 궁금합니다.**  - 커버리지에 대한 % 데이터를 아는 것이 중요하지만, 개발자 입장에서는 전체 100% 중 커버가 안된(미스된 부분) 부분이 어딘지를 아는 것이 중요할 것 입니다.  - JACOCO 데이터를 파싱해서 일부만 저장하는 것 보다는 html 폴더(JACOCO report 전체)를 가져와서 페이지로 보여주는 것도 좋아보입니다. 최대한 오픈소스를 활용하세요.  **5. Maven 플러그인을 통해 MongoDB에 junit, jacoco xml에 대한 데이터를 파싱해서 전부 저장하려고 합니다. MongoDB에서 미리 프로젝트와 테스트, 커버리지에 대한 데이터 사이의 관계를 만들어야할까요? 아니면 정제하면서 관계를 지정해주어야할까요?**  - 저희 서비스에서는 MongoDB(DATA server)에서부터 job id를 지정해서 연결해주었습니다.  **6. Spring에서 maven package 시, test가 fail 되면 build가 안되는데 Pass Rate가 100%가 안될 때, 어떻게 배포를 하는건지 궁금합니다.**  - 원칙은 100%가 아니면 배포 불가합니다. 하지만 대부분은 테스트 코드의 오류이고 실제 시스템의 오류는 아닌 경우가 대부분이므로 테스트 코드 상의 오류라고 판단하여 배포하는 경우가 많습니다.  **7. 여러 커버리지 중에서 중점적으로 관리하는 커버리지의 종류와 그 이유가 궁금합니다. 또, 몇 %를 하한선으로 두고 있는지 궁금합니다.**  - 현재 SDS에서는 라인 커버리지는 70%, 브랜치 커버리지는 60%를 기준으로 잡고 있으나 개발 팀, 프로젝트마다 다릅니다.  - 예를 들어 생명과 관련된 일과 같은 경우에는 커버리지가 무조건 100%여야 하겠습니다.  - 레거시 코드(오래된 코드)들은 라인 커버리지 70%도 만들 수 없는 솔루션도 있습니다.  - 일부 개발자들은 라인을 거쳤다고 테스트가 진행된 것이 맞을까? 하는 의문을 가지는 개발자들도 있습니다. 테스트 결과가 맞고 틀리고가 아닌 테스트가 안되는 곳을 찾는 목적으로 커버리지, 테스트 결과를 보려고 하는 것입니다.  - 멘토님께서 추천해주신 사이트  구글의 커버리지  (<https://testing.googleblog.com/2020/08/code-coverage-best-practices.html>)  커버리지 권장치  (<https://testing.googleblog.com/2014/07/measuring-coverage-at-google.html>)  단위테스트 이점  (<https://blog.pragmaticengineer.com/unit-testing-benefits-pyramid/>)  **8. 최근 한 기업의 컨퍼런스에서 코드 커버리지 100%에 도전하고 실제로 성공해서 관리를 하고 있는데(Instruction length: 80,000, Production Code Line: 4,500, Test Code Line: 8,000) 100% 달성 시도를 한 적이 있었는지, 실패했다면 그 이유가 무엇인지 궁금합니다.**  - 커버리지 100%를 달성하는 것은 의미가 없다고 생각합니다.  - 투자 대비 효율이 나오지 않습니다.  **9. 업무 시간 중 프로덕션 코드 작성 시간과 테스트 코드 작성 시간의 비율이 궁금합니다.**  - 테스트 코드에 미숙한 개발자들은 개발에 투자하는 만큼 테스트 코드 작성에도 시간이 걸립니다.  - 일반적으로는 테스트 코드를 작성하는 시간이 30~50% 정도 소요됩니다.  [추가 말씀]  단위 테스트의 목적  1. 리팩토링의 목적(단순화, 구조화, 모듈화)  2. 설계서 역할  3. 디버깅  4. 개발의 일부입니다.  **[멘토 전달사항]**  시스템에 대해 고민을 더 많이 하셔야 할 것 입니다.  **[건의사항]**  없었습니다. | | | |
|  | | | |
|  | | | |